

Toffoli Gate Implementation using the Billiard Ball Model

Hadi Hosseini  
Faculty of Computer Science  
University of New Brunswick  
Email: h.hosseini@unb.ca

Gerhard W. Dueck  
Faculty of Computer Science  
University of New Brunswick  
Email: gdueck@unb.ca

Abstract— In this paper we review the Billiard Ball Model (BBM) introduced by Toffoli and Fredkin. The analysis of a previous approach to design reversible networks based on BBM is shown to ignored physical realities. We prove that some logic function cannot be realized without additional control balls. For example, to realize the logical OR operation, at least three control balls are needed. We show how reversible Toffoli gates can be constructed with this model. Finally, a Toffoli gate module is proposed that can be used in a cascade of gates and thus implement arbitrary reversible functions.

I. INTRODUCTION

In the current digital era energy efficiency is becoming more and more important in our computer-based world, as most of the services are being provided using various types of computational machines such as workstations, personal computers and etc. One potential way to address this issue is through non-dissipative computing which lies beyond the Landauer’s principle [1] [2]. It states that any logically irreversible manipulation of information, such as the erasure of a bit or the merging of two computation paths, must be accompanied by a corresponding entropy increase in non-information bearing degrees of freedom of the information processing apparatus or its environment [3] [4]. Any computational process, which does not consume energy, is (at least to some close approximation) reversible, i.e., time-invertible, meaning that a time-reversed version of the process could exist within the same general dynamical framework as the original process. Thus, this is also called reversible computing.

The billiard-ball model was first introduced by Edward Fredkin and Thomas Toffoli [5] in an attempt to design a reversible method of computing logic functions using elastic collisions involving balls and fixed reflectors. Intuitively, computer hardware is assumed as a container with suitable shape and computer software considered as proper initial condition of balls and collisions between them. The rules for this model are identical to those that underlie the classical kinetic theory of perfect gases where the balls are interpreted as gas molecules and the reflectors as sections of the container’s walls. Essentially, billiard-ball computing provides two types of interacting objects, ”balls” and ”walls”. The basic insight is that a location where balls may or may not collide acts like a logic gate: we get a ball coming out at certain places only if another ball did not redirect it from it’s trajectory.

The billiard ball machine is a model that roughly simulates the action of ”billiard balls” colliding with obstacles and each other on a two-dimensional playing field. If some way of indicating direction of the balls is added, the Billiard Ball Model (BBM) essentially becomes a switching circuit. It then becomes Turing-complete if an infinite number of balls/obstacles is allowed [6]. When the BBM is initialized appropriately, the sequence of states that appear at successive integer time-steps is equivalent to a discrete digital dynamics [7].

In this paper, we introduce the billiard ball model and the possible gates that are developed as basic gates in attempt to create reversible circuits. In section III we will discuss a proposed model by Perkowski and argue on some of the flaws in this model. After that, in section IV-A, we show a possible solution or the OR gate. Section IV-B discusses our simulation environment and finally in sections V we will show a universal gate built using the billiard ball model and modularize it to be able to create any reversible logic function by using balls and mirrors.

II. BILLIARD BALL MODEL

In the BBM, the environment is considered to be a two-dimensional grid and hard balls of radius \( l/\sqrt{2} \) traveling along the grid’s principal directions at the velocity of one unit of space per time unit. Clearly, the presence or the absence of a ball at a given point of the grid can be interpreted as a binary variable, taking on a value of 1 for ”ball” or 0 for ”no ball” at each integral value of time, and the place where signals intersect acts as a logic gate, with different logic functions of the inputs coming out at different places.

All balls should be synchronized in the system in order to form standard collisions within the billiard ball model. Consequently, balls should spend the same amount of time in the module so they can result simultaneous collisions in the output. The fundamental rules of billiard ball computing are the exact laws used in Newtonian kinematics. Movement and collision of billiard balls are assumed to be completely non-dissipative in an ideal elastic environment. However, physical realities such as friction and colliding one ball into a wall or another ball cause considerable amount of energy loss in the system.
For most people, the word computer will bring up images of a box with CRT and a keyboard. But there are lots of other computing mechanism one can think of [8]. The idea of designing a billiard ball computer in a large scale is considered to be ambitious and unachievable with today’s technologies. However, future nanotechnologies may equip us to use molecules and atoms in particles and apply controlled forces to them to change the arrangement of them and make them move in any direction to cause predesigned collisions. Then, we will be able to observe the outputs and make logic gates at micro or nano scales. A molecular machine has been defined as a discrete number of molecular components that have been designed to perform mechanical-like movements (output) in response to specific stimuli (input). It is often applied more generally to molecules that simply mimic functions at the macroscopic level [9] [10]. Eventually, billiard ball computing might be useful to build electric-free logic circuits in very small sizes in nanotechnology or quantum computing [11].

Presence or absence of "a ball" in specific location at specific time is interpreted as desired result. Since billiard balls are indistinguishable, it is possible to create an "interference-free" crossing, where two balls cross paths, using only mirrors. All collisions are elastic, in other words, no energy is lost, so the billiard ball computer can run forever.

The fundamental interaction gate in BBM was proposed by Toffoli and Fredkin in order to build a physical model of boolean logic gates. The interaction gate and the switch gate are simple BBM gates that are able to realize limited functions in a reversible way [5], [12]. The interaction gate is the conservative-logic primitive of billiard ball model. Figure 1 shows the realization of the interaction gate in BBM.

Another gate was suggested by Ressler [13], it realizes the conditional routing of one data input by another control ball. As it is shown in Figure 2.a, ball A controls the input ball of B and changes the output.

Having a control ball in the system enables us to build a NOT gate by just changing one of the inputs of the interaction gate to a control ball and as a result come up with a negation gate as shown in Figure 2.b. We use the term “control ball” to indicate that the ball will always be present – unlike a variable ball, which is only present if the corresponding variable has the value of 1.

### III. CRITIQUE ON PERKOWSKI’S MODEL

Perkowski [14] introduced a method to create Boolean gates based on the definition of the billiard ball model in order to make reversible logic circuits. He showed that any gate can be created using BBM primitive interaction gates and its reverse function. In the Figure 3, a full adder has been created based on this idea. The figure shows that by having two interaction gates and three inverse interaction gates we are able to create a full adder.

This seems to be a reasonable sketch to build a full adder; however, there are some issues regarding the mechanical constraints of this model that need to be addressed. The billiard ball model is a physical representation of logic gates. Thus, when building any circuit with billiard balls all physical contrains need to be respected. Unlike electrical signals, billiard balls are not able to have fan-in or fan-out in the terminals (i.e. inputs and outputs.)

By inverting the interaction gate and creating its truth table, two OR gates with disjoint inputs of balls will be formed. If there is a ball in AB as Z1 there will be another ball coming through the last output as Z4. Because A’B as Z2 and AB’ as Z3 are disjoined inputs, just one of them may be active in the input to form the output of invert interaction gate.

The truth table in Figure 4 shows that ORing the first with the third input and ORing the second and the last input will result in getting A in one of the outputs and B in the other one respectively. However, in order to build an OR gate using billiard balls, they need to be redirected in the same trajectory – due to the physical constraints, this is impossible (the proof is presented in the next section.)
IV. IMPLEMENTING AN OR GATE

Due to the physical limitations of billiard balls, connecting disjoint outputs of the interaction gate to the inputs of an invert interaction gate is not feasible to OR the outputs. Moreover, designing a logical disjunction with billiard balls is not a straightforward process due to the fact that directing two different balls with different directions into a single path needs to use a mirror in the interference point which always redirect the other ball to the opposite direction.

Theorem 1: Given a ball A or a ball B on different trajectories, both cannot be directed to a single trajectory at the same time.

Proof: Assume ball B should be on the same trajectory of ball A. To place B in the trajectory of A, a mirror has to be placed in the trajectory of A which will divert A to another trajectory. Alternatively, a collision with a control ball can be used, however, the control ball will direct A from its initial trajectory.

According to Theorem 1 the balls A'B and AB' from the interaction gate (see Figure 1) cannot be placed on a single trajectory. It is thus necessary to add control balls. A possible solution is shown next.

A. Control Balls as Solution

As discussed in section IV, unlike logical conjunction, realizing a logical disjunction function is not a trivial process in BBM. Having a logical conjunction and negation gates establishes a set of functions consisting of AND and NOT functions. It is clear that this set of functions is universal (being able to form any other logical function) as it holds all the properties required such as; not being subset of 0-preserving, 1-preserving, self-dual, monotone increasing and linear functions.

Theorem 2: A BBM implementation of the OR function requires at least three control balls.

Proof: Using Theorem 1 we know that it is impossible to redirect two balls into the same trajectory at a given time. Also, Demorgan’s law enables us to use AND and NOT gates to create an OR gate. Every OR gate with two inputs can be represented by an AND gate and three NOT gates to invert the logic values before and after applying logical conjunction (two for inverting each variable and one to invert the result output of the AND gate). Hence, one control ball for each negation is needed i.e. three control balls to implement an OR gate.

As it is illustrated in the Figure 5 and stated in 2, linking three interaction gates and having three control balls controlling the outputs of each, makes it possible to physically model an OR gate.

B. Simulating the BBM gates

The simulation environment we have developed is a simple JAVA Applet consisting of balls and walls, as the main elements, moving in four directions North, South, East, West in a gridline map. Each color in the environment represent a direction Blue, Green, Red and Yellow represent North, South, East and West respectively. If the direction of a ball changes (by colliding to another ball or meeting a mirror on its way), then the color of ball will change with respect to its new direction.

Applying the same method and making use of the outputs created by basic interaction gate enables us to design an XOR gate. In this gate, three control balls are used to make the redirections and control the output. For simulating billiard balls and collisions, an application by Lin [15] has been extended to meet our needs. The computer program is used to simulate the behavior of billiard balls in a physical environment. Figure 6 shows a screen shot of an OR gate implemented in our simulation environment. The simulation application (implemented as an Applet) and some video captures are available at http://people.unb.ca/~e2nu0/bbm.
wants to arrange mirrors and balls in the gridline. Nevertheless, there are some arrangements of balls and walls that are impossible to act as the desired layout. This is due to the space and timing limits. Clearly, some designs are more efficient than others.

V. Modeling Toffoli Gate in BBM

The Toffoli gate, invented by Tommaso Toffoli [16], is a universal reversible logic gate, which means that any reversible circuit can be constructed from Toffoli gates. It is also known as the "controlled-controlled-not" gate, which describes its action.

The Toffoli gate is a universal gate. That is, any logic gate can be created just by having several Toffoli gates connected with each other. In Figure 7, a structure of a Toffoli gate represented in BBM is been shown. The idea is using one interaction diagram, as described in II, and connect it to an exclusive disjunction module, delineated in section IV-A, and construct a new hierarchy of gates which result in a Toffoli gate.

Any interaction gate in BBM has two inputs and four possible outputs. Two balls in the input of each gate emulate two electrical signals entering a gate, however having a physical implementation of gates originates in disjoint results in the output meaning that there is always just one output enabled at any time. Hence, different inputs to the interaction gate produce disjoint values in the output. Having A and B as inputs to the interaction gate, the set of possible outputs consists of AB, A'B, AB', all disjoint values. Considering X, Y and Z as inputs of the Toffoli gate, by definition we need to produce three outputs representing X and Y and also X \oplus Y\odot Z on the last line.

As discussed earlier, in modeling Toffoli gate using billiard balls we are facing some limitations due to the physical nature of collisions and the fact that balls, as oppose to electrical signals, are not separable into two or more branches i.e. having fan-in or fan-out is not possible while working with physical models. Also, as X and Y are used in the first interaction gate it is impossible to reuse them again in the output because of the uncertainty of their presence in the output (it is indeterminate to know whether any of the input balls will, for sure, be present in the output in the same trajectory). To be able to have two instances of each ball in the input, we make use of an extra interaction gate for X and Y, and also add an always-true control ball as one of the inputs of the added interaction gates. Therefore, whenever X and Y are present in the input (indicating logical "1"), there are two outputs enabled representing X (or Y). Similarly, if X is not present in the input (indicating "0") the aforementioned outputs will hold a value of 0 (no ball). Since we have produced two instances of X and Y, now it is possible to redirect one of each directly to the output as two desired outputs of the Toffoli gate. By redirecting one of the instances of X and Y (using mirrors) into another interaction diagram, a gate with the output set consisting of XY, X'Y, XY', XY will be created. Note that two members hold the same value of XY but with different trajectory that is the purpose of showing both in the output set.

The remaining output line in the Toffoli gate is the output showing the exclusive disjunction of an input with the logical conjunction of two other inputs. Therefore, by directing the XY output of the latter interaction gate into another interaction gate (as shown in Figure 8) we end up with a set of output consisting XYZ, (XY)'Z , XY(Z)'. Knowing that XOR gate with inputs A and B implements the logical expression AB' + A'B, essentially all we need is to get these outputs, direct them to an appropriate track, using mirrors, to apply a logical disjunction and have an exclusive OR in the output. As we discussed in section IV, Theorem 1 shows that an OR gate cannot be created by just redirecting balls into a single path, therefore, we need to use the OR gate created using three control balls. By adding any desired mirrors and balls into the structure, we are able to physically simulate the interaction between them.

A. Toffoli Module

As shown in the previous section, we have designed a Toffoli gate using billiard balls. In this section we introduce a BBM module which can be used to create any other logic circuit. Any logic circuit can be represented in a series of cascading gates created by connecting number of Toffoli gates. As it is shown in Figure 10 this module consists of three input lines and three output lines. Also there are five control balls, two balls for duplicating two of the inputs X and Y and as declared in Theorem 2 three other balls to represent NOT gates using in the OR part of the gate.
Finally, to form a new logical circuit we can simply add desired number of Toffoli modules and connect the outputs to the appropriate inputs. We will need to make use of delay gates in order to delay the control balls used in the second and third level of the circuit and use mirrors in the way to direct each ball to the appropriate trajectory that it should follow.

Representing any desired logical function in BBM will cause number of Toffoli gates connecting to each other using the modules shown in Figure 10. As discussed earlier, each Toffoli module needs five control gates and number of mirrors between the outputs to redirect the final results (three output lines) to the inputs of the next module.

Space and time are the most crucial factors in designing logical circuits using the BBM. The former needed to build up a logic circuit in BBM may vary depending on the number of Toffoli gates and control balls needed to form the structure of the final circuit. However, some optimizations in the layout can be added to decrease the needed area avoiding undesired collisions which may lead to a chaos in the system. The latter, on the other hand, is one of the most important aspects which should be set precisely. Due to the synchronized nature of the BBM gates, all the control balls in each level should be set into delay before entering the respective modules.

In addition to space and time, the number of control balls plays an important role in building and using billiard ball models. More control balls necessitate more space in for circuits and more complicated synchronization between different gates and control balls. Additionally, control balls require energy.

VI. CONCLUSION

In this paper, we reviewed the Billiard Ball Model suggested by Fredkin and Toffoli, which is based on elastic collisions between balls and walls (mirrors) without any energy loss. The fundamental BBM gates such as the interaction gate and the switch gate are explained — they from the basis for the functionality of this model. As a physical model of computation logic, its physical constraints can not be ignored when dealing with implementation of logic functions. We critiqued Perkowski’s attempt to create a full adder function using interaction gates and inverted interaction gates. We showed the inherit difficulty of constructing an OR gate. Based on DeMorgan’s law we proposed an OR implementation with three control balls (Theorem 2). By building a universal function using billiard balls, we are able to create any logic circuit in BBM and therefore. As a proof of concept, we designed and simulated a Toffoli gate just using billiard balls. To test our designs, we developed a simulation environment in JAVA to show the movements and collisions of billiard balls in plain surface while moving toward four possible directions North, South, West, East. Eventually, by introducing a Toffoli module it is, in fact, clear that any logic function can be created just by using some balls as inputs, some mirrors as redirecting walls and some control balls.

REFERENCES